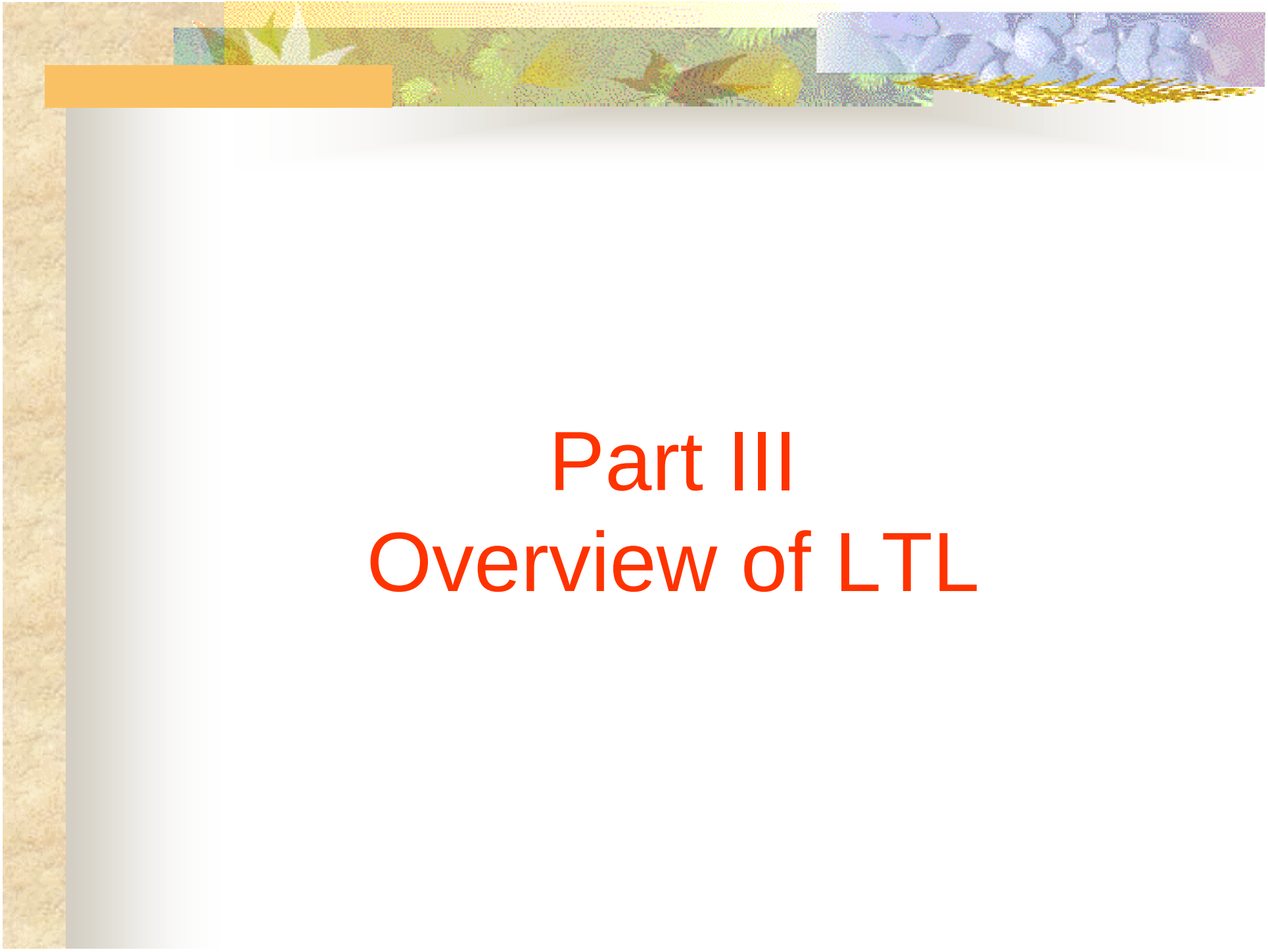
Introduction to Spin and Promela

Sagar Chaki

CMU

Roadmap

- Historical perspective
- Overview of Spin
- Overview of Promela
- Simulation with Spin
- Overview of LTL
- Verification with Spin



Part III
Overview of LTL

Basic concepts

- Set of propositions: P
  - P = {a,b,c}
- Infinite trace τ over P
  - $t_0, t_1, t_2, \ldots$
  - t_i subset of P for i 0
  - τ_i denotes infinite trace $t_i, t_{i+1}, \ldots$
  - {b},{a,c},{},{a,b,c},{a}, …
- Set of all infinite traces over P: $\Gamma(P)$

LTL syntax

- ϕ := p proposition
  - | true
  - | false
  - | (ϕ)
  - | ϕ binop ϕ
  - | unop ϕ

LTL syntax

■ unop :=		[]	always (G)
	|	<>	eventually (F)
	|	X	next time
	|	!	logical negation
■ binop :=		U	strong until
	|	&&	logical AND
	|	||	logical OR
	|	->	implication
	|	<->	equivalence

LTL semantics

- An infinite trace τ (over P) either satisfies or does not satisfy an LTL formula ϕ (over P)
  - Satisfaction denoted by $\tau \models \phi$

LTL semantics

- Given an infinite trace $\tau = t_0,t_1,t_2, \ldots$ and a LTL formula ϕ we can decide if

  $\tau \models \phi$ depending on the structure of ϕ

  - $\tau \models p$ iff p belongs to t_0
  - Always $\tau \models$ true
  - Never $\tau \models$ false
  - $\tau \models (\phi)$ iff $\tau \models \phi$

LTL semantics

- $\tau \models [] \phi$ iff $\tau_i \models \phi$ forall i 0
- $\tau \models <> \phi$ iff exists i 0 s.t. $\tau_i \models \phi$
- $\tau \models X \phi$ iff $\tau_1 \models \phi$
- $\tau \models !\phi$ iff NOT($\tau \models \phi$)

- $\tau \models \phi_1 U \phi_2$ iff exists i 0 s.t. $\tau_j \models \phi_1$ for 0 $j < i$ and $\tau_i \models \phi_2$

LTL semantics

- $\tau \models \phi_1 \,\&\&\, \phi_2$ iff $\tau \models \phi_1$ AND $\tau \models \phi_2$
- $\tau \models \phi_1 \,||\, \phi_2$ iff $\tau \models \phi_1$ OR $\tau \models \phi_2$
- $\tau \models \phi_1 \rightarrow \phi_2$ iff $\tau \models \phi_1$ IMPLIES $\tau \models \phi_2$
- $\tau \models \phi_1 \leftrightarrow \phi_2$ iff $\tau \models \phi_1$ IFF $\tau \models \phi_2$

LTL semantics

- An LTL formula can also be looked at as the set of infinite traces that satisfy it
  - Note the striking similarity with the case of regular expressions and finite strings over an alphabet Σ
  - If S is the set of traces that satisfy ϕ then $\Gamma(P)\backslash S$ is the set of traces that satisfy $!\phi$

Examples

- {a},{b},{a},{b}, …
- []($a \rightarrow X b$) []($a \leftrightarrow X b$)
- []($b \rightarrow X a$) []($b \leftrightarrow X a$)
- []($a \rightarrow (b \, U \, a)$)
- []($a \rightarrow (a \, U \, b)$)

Things to remember

- Every LTL formula represents the set of infinite traces which satisfy it

Buchi automata

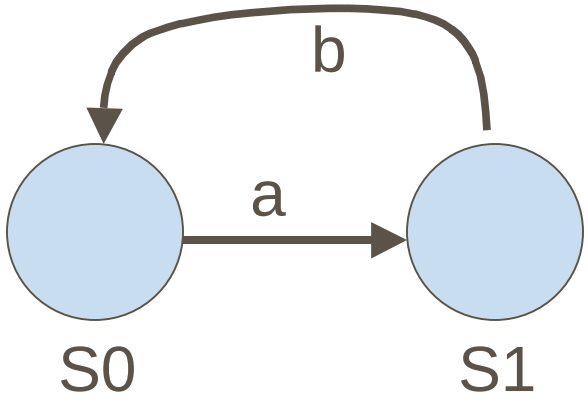
- Recall the similarity of LTL with regular expressions
- Regular languages are accepted by finite automata
- Are there automata for LTL
  - Turns out there are
  - They are called Buchi automata

Buchi automata

- Fix an alphabet Σ
- Buchi automaton is a 4-tuple : $<Q,I,\delta,F>$
  - Q : set of states
  - I : initial state
  - δ : transition relation: subset of $QX\ \Sigma\ XQ$
  - F : set of accepting states

- In our case Σ is 2^P

Example



I = S0 , F = {S0}

Buchi automata

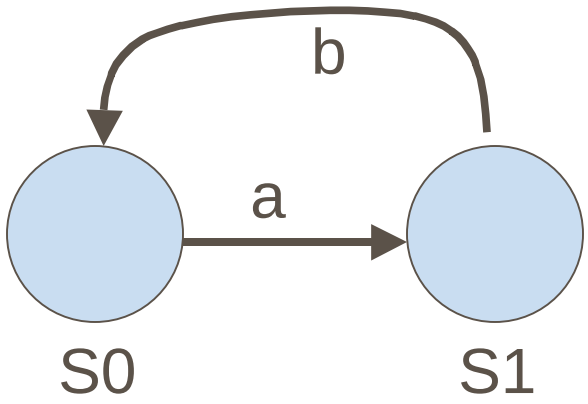
- A run is an infinite sequence of state $s_0,s_1,s_2, \ldots$ such that
  - $s_0 = I$
  - Exists a_i º Σ s.t. (s_i,a_i,s_{i+1}) º δ for i 0
- A run is an accepting run iff it visits some accepting state infinitely often

Buchi automata

- Given a run $\sigma = s_0,s_1,s_2, \ldots$a trace $\tau = \tau_0,\tau_1,\tau_2, \ldots$is said to correspond to σ iff (s_i,τ_i,s_{i+1}) º δ for i 0
- The language of a Buchi automaton is the set of traces corresponding to its accepting runs
  - LTL also corresponds to a set of traces

Example

- Traces : {{a},{b},{a},{b},…}



I = S0 , F = {S0}

Cute results

- BA are closed under complementation
  - For every Buchi automaton A there exists another Buchi automaton B such that the language of A is the complement of the language of B
  - Here complement means set difference from Γ(P)

Cute results

- BA are closed under intersection and union
- Checking if the language of a Buchi automaton is empty is decidable
  - Can check if the langauges of two Buchi automata have a non-empty intersection

Cute results

- Buchi automata are not always determinisable
  - There exists a non-deterministic Buchi automaton A such that there is no deterministic Buchi automaton with the same language as A
  - Non-deterministic means some states have multiple outgoing transitions with same label

Cute results

- The set of languages accepted by Buchi automata is called the set of

  ω-regular languages

  - This is a strict superset of LTL
  - Every LTL corresponds to some BA
  - There exists a BA whose language does not correspond to any LTL formula

Things to remember

- Every LTL formula represents the set of infinite traces which satisfy it
- Every LTL formula has a corresponding Buchi automaton

Kripke structure

- Is a 4-tuple : <S,I,δ,L>
  - S : set of states
  - I : initial state
  - δ : transition relation: subset of S X S
  - L : labeling function : S -> 2^P

Computations and Traces

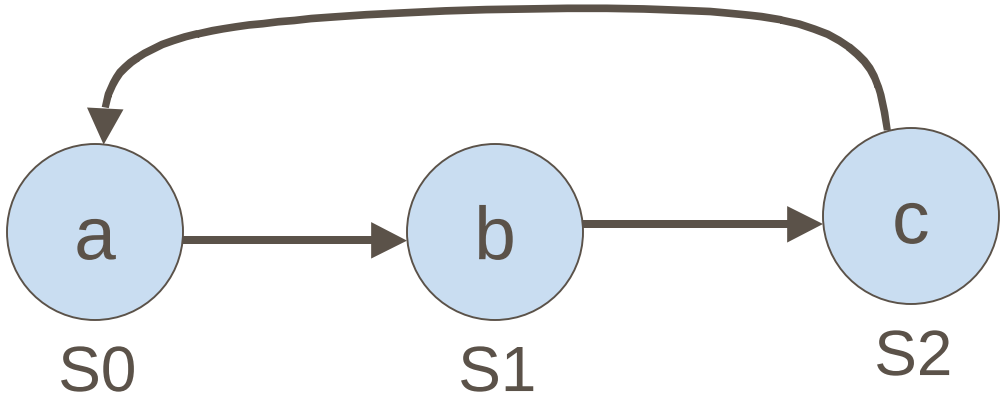
- A computation is an infinite sequence of states $s_0, s_1, s_2, \ldots$
  - $s_0 = I$
  - $(s_i, s_{i+1}) \circ \delta$ for i 0
- Given a computation $s_0, s_1, s_2, \ldots$ the corresponding trace is

  $L(s_0), L(s_1), L(s_2), \ldots$

Kripke and Buchi

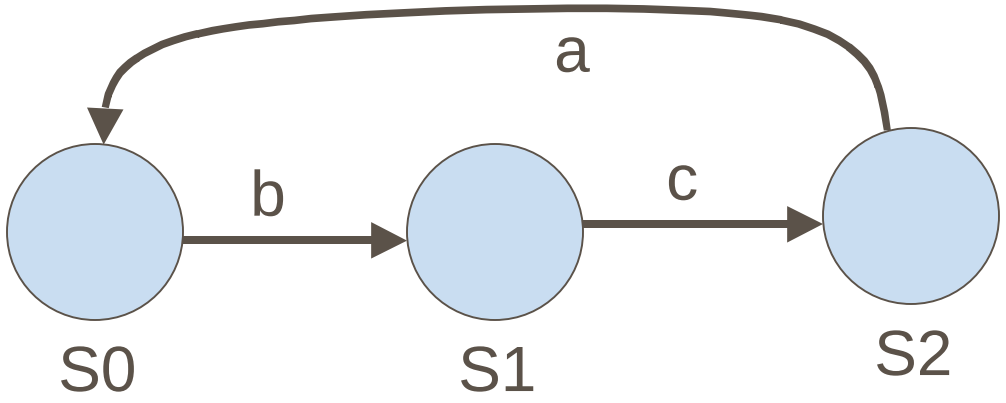
- Kripke looks different from Buchi
  - Labels are on states not transitions
  - No accepting states
  - Nevertheless …
- For every Kripke structure K there exists a Buchi automaton which accepts exactly the set of traces corresponding to computations of K

Kripke to Buchi

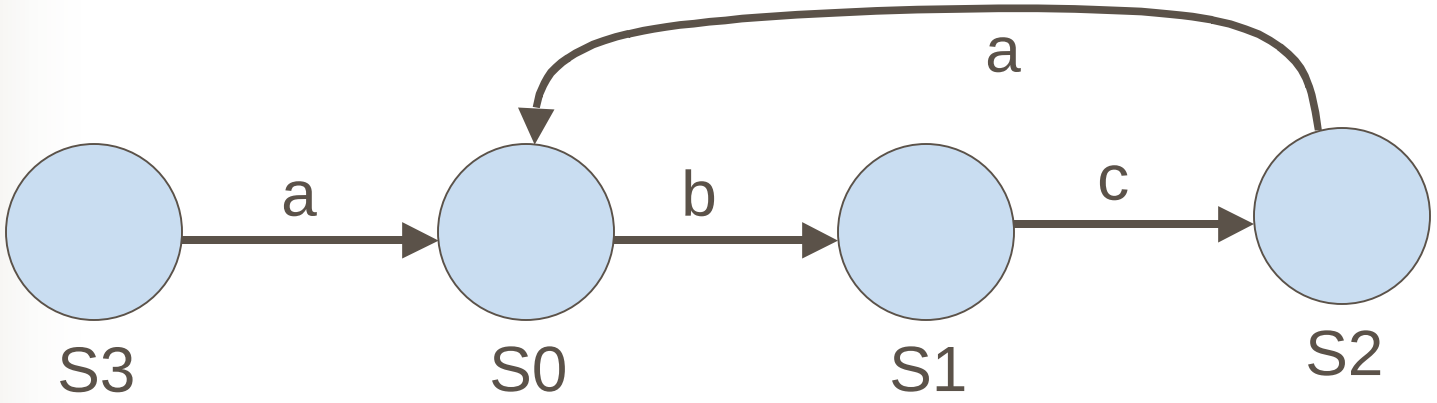


I = S0

Kripke to Buchi



Kripke to Buchi



I = S3, F = {S0,S1,S2,S3}

Things to remember

- Every LTL formula represents the set of infinite traces which satisfy it
- Every LTL formula has a corresponding Buchi automaton
- Every Kripke structure has a corresponding Buchi automaton

What do we really want?

- Kripke (M) ² LTL (ϕ)
- Traces of M contained in traces of ϕ
- Language of Buchi of M contained in language of Buchi of ϕ
- Language of Buchi of M has empty intersection with language of Buchi of $!\phi$
  - This is decidable

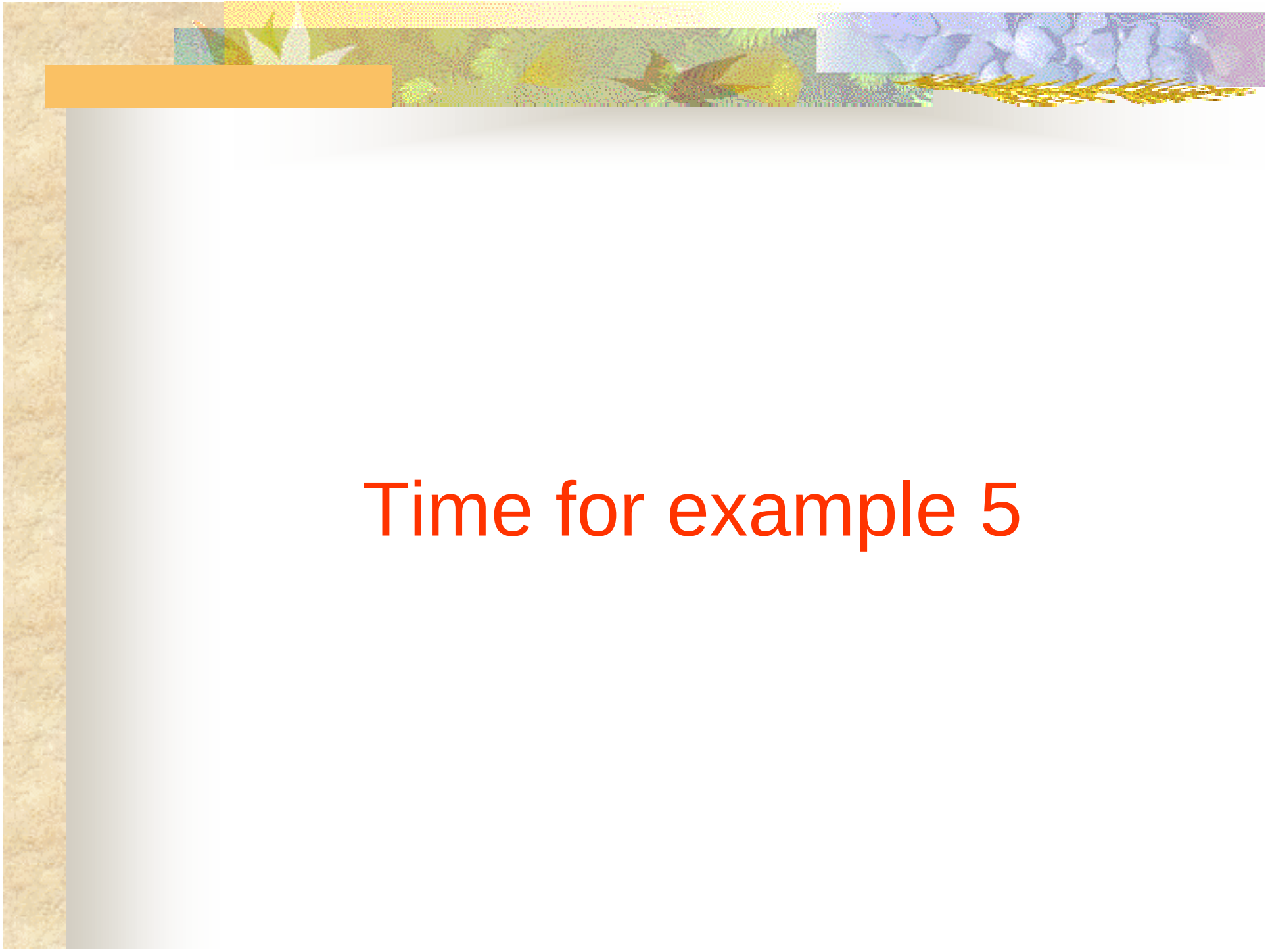
LTL model checking

- Two ways to do it
- Convert Kripke to Buchi
  - Convert claim (LTL) to Buchi
  - Check language inclusion

    OR
  - Convert ~Claim (LTL) to Buchi
  - Check empty intersection

What Spin does

- Checks non-empty intersection
  - Requires very little space in best case
- Works directly with Promela
  - No conversion to Kripke or Buchi
- Must provide Spin with negation of property you want to prove


Time for example 5

References

- http://cm.bell-labs.com/cm/cs/what/spin/
- http://cm.bell-labs.com/cm/cs/what/spin/Man/Manual.html
- http://cm.bell-labs.com/cm/cs/what/spin/Man/Quick.html